Discovering Cardiac Action Potential Model Equations Using Sparse Identification of Nonlinear Dynamics

Cole S Welch[1], Elizabeth M Cherry[1]

[1] Georgia Institute of Technology, Atlanta, GA, USA

Abstract

Many models of cardiac action potentials (APs) have been developed, but identifying appropriate equations and parameter values to match particular datasets remains a challenge. To reproduce cardiac AP data, we consider the use of a data-driven approach, Sparse Identification of Nonlinear Dynamics (SINDy). SINDy is a sparse regression method that uses a set of chosen candidate functions to produce a differential-equations model that fits the provided data. Terms with small coefficients are iteratively discarded to reduce model complexity while maintaining an accurate fit. We analyzed SINDy's effectiveness in fitting synthetic AP data from two-variable models with polynomial terms, including the FitzHugh-Nagumo model (FHN), its cardiac variant that avoids hyperpolarization (FHN-c), and two additional cardiac-modified FHN models that can display complex dynamics. We found that SINDy could effectively reproduce the equations for each model, with the cardiac variants displaying greater sensitivity to parameter and optimizer choice than the baseline FHN model. Finally, we tested the ability of SINDy to handle the introduction of time-dependent stimulus currents, including identification during alternans dynamics. Overall, SINDy shows promise as an approach for identifying differential equations models to match cardiac AP data while balancing model complexity and accuracy.

1. Introduction

Since the development of the first differential-equations-based cardiac action potential model by Noble [1], models of cardiac cell electrophysiology have been created primarily in a mechanistic fashion. In this framework, measurements are used to formulate mathematical descriptions of ion channel dynamics, often following the formalism of Hodgkin and Huxley [2], and are integrated with intracellular ion concentration dynamics to describe the behavior of the action potential. The most detailed models can have dozens of ordinary differential equations (ODEs) with hundreds of parameters, leading not only to concerns about model identifiability but also to difficulty understanding the mechanisms underlying observed model properties and behavior.

Recently, data-driven approaches for discovering equations that describe dynamical systems have been postulated. Such methods may offer opportunities for developing simpler model forms that directly fit specific experimental or clinical data. However, it has historically been difficult to obtain interpretable results from machine-learning methods. In neural networks, for instance, the trained model consists of a sum of weighted activation functions containing a large number of terms, which cannot easily be translated into a human-readable set of equations [3]. To address this problem, Brunton and Kutz et al. introduced Sparse Identification of Nonlinear Dynamics (SINDy) in 2016. Their approach utilizes a priori understanding of the physics behind a problem to inform which functional forms could feasibly show up in a system of ODEs describing the dynamics of interest [3].

In this paper, we first demonstrate the use of SINDy to recover action potentials from simple cardiac models and study the effect of algorithmic choices. We then test the ability of our cardiac SINDy implementation to reproduce paced dynamics including alternans, in which action potentials vary in duration.

2. Methods

SINDy: SINDy utilizes a set of candidate functions Θ weighted by a coefficient matrix Ξ in order to fit a given dynamical system of the form $\dot{\mathbf{X}} = f(\mathbf{X}(\mathbf{t}))$ to data, which is described by the following equation [3]:

$$\dot{\mathbf{X}} = \mathbf{\Theta\Xi}. \tag{1}$$

The SINDy fit is accomplished by applying a least-squares fit to the data combined with an L1 regularization term (weighted by the coefficient α in the formulation below), which penalizes large coefficients [3]:

$$\min_{\Xi} \|\dot{\mathbf{X}} - \mathbf{\Theta\Xi}\|_2^2 + \alpha\|\mathbf{\Xi}\|_1. \tag{2}$$

The SINDy framework was implemented via the Python library PySINDy, which offers flexibility in defining function libraries, selecting an optimizer, and integrating with other machine-learning methods [4].

Cardiac Electrophysiology Models: For this proof-of-concept work, test data were generated from phenomenological two-variable models that have been used to describe cardiac action potentials. As simple test cases, the FitzHugh-Nagumo (FHN) model [5] (Eqns. 1a and 2a in Table 1, with μ fixed to 1) was used, along with a cardiac-inspired FHN modification (FHN-c, Eqns. 1b and 2a) that prevents hyperpolarization [6]. For additional test cases, we generated scenarios using the 4-parameter (Eqns. 1a and 2b) and 7-parameter (Eqns. 1a and 2c) Velasco-Fenton models [7] (VF-4 and VF-7, respectively), including an alternans case with the VF-4 model.

Table 1. Model equations.

Label	Equation
1a	$\dot{u} = \mu u(u-\alpha)(1-u) - v$
1b	$\dot{u} = u(u-\alpha)(1-u) - uv$
2a	$\dot{v} = \epsilon(\beta u - \gamma v - \delta)$
2b	$\dot{v} = \epsilon(u(\beta - u) - v)$
2c	$\dot{v} = \epsilon((\beta - u)(u - \gamma) - \delta v - \theta)$

When stimuli were applied, square-wave excitations with durations of 5 time units and magnitudes of 0.12 normalized voltage units were added to the voltage variable for each of the four models to generate action potentials.

To generate the data, SciPy's `odeint` integrator, an adaptive, variable-method solver based on the LSODA Fortran library [8], was used with a maximum step size of 0.1 for 2,000 time units. A non-autonomous function was added to the voltage variable u before solving to handle stimuli. The resulting dependent variables u and v, along with the independent time values t, were passed to PySINDy instances. When a single action potential was fit from a super-threshold initial condition (no stimulus), either default or degree-3 polynomial function libraries were used. When stimuli were applied, custom libraries containing two separate sublibraries were used, one for functions of the voltage and/or recovery variables and one containing only time-dependent terms (including only the stimulus voltage term, the timing of which is known from the dataset), to allow the PySINDy instance to recognize the functional forms of these user-defined terms.

Each PySINDy instance was tested across a variety of optimizers, including STLSQ (Sequentially Thresholded Least Squares), SR3 (Sparse Relaxed Regularized Regression), and SSR (Stepwise Sparse Regression). STLSQ and SR3 require manually specified threshold parameters in addition to the L1 regression coefficient α, while SSR only requires a value for α. The first two methods discard any terms with coefficients below the threshold at once, while SSR iteratively discards a single term with the lowest magnitude coefficient, causing differences in convergence behavior depending on the use case.

Python code implementing the procedures described in this section can be found in the GitHub repository included in the references [9].

3. Results

As a first test scenario, we considered fitting single action potentials produced by the four models using initial conditions above the threshold of excitation, such that the additional complexity associated with adding the stimulus current in SINDy was not necessary. The results of fitting with various feature libraries and optimizers for the FHN, FHN-c, VF-4, and VF-7 models are shown in Table 2. For the FHN model, the correct equations could be identified with any of the three chosen optimizers as long as the function library was restricted to degree-3 polynomials. However, using the STLSQ and SR3 optimizers required more careful threshold parameter selection, a disadvantage which the SSR optimizer did not have because SSR discards one term at a time until the desired model complexity is reached. Thus, it requires no fixed threshold parameter, and instead only requires tuning the L1 regularization weighting coefficient α. With the default PySINDy function library, which includes a wider variety of terms, none of the three optimizers yielded a successful identification; for brevity, only the SSR result is shown.

Table 2. Identification success outcomes for SINDy. Optimizers tested were STLSQ, SR3, and SSR. "Poly3" represents a polynomial library containing all possible combinations of two variables with at most degree three, while "Default" is the default PySINDy library (which contains a variety of polynomial, trigonometric, and exponential terms). Sensitivity is reported with respect to threshold or regularization parameter.

Model	Library	Optimizer	Success	Sensitivity
FHN	Poly3	STLSQ	Yes	High
	Poly3	SR3	Yes	High
	Poly3	SSR	Yes	Low
	Default	SSR	No	N/A
FHN-c	Poly3	STLSQ	No	N/A
	Poly3	SR3	No	N/A
	Poly3	SSR	Yes	Low
	Default	SSR	No	N/A
VF-4	Poly3	SSR	Yes	Low
VF-7	Poly3	SSR	Yes	Low

For FHN-c, identification was also unsuccessful without

SINDy fit coefficients:

$$u' = -0.19906u + 1.19714u^2 - 0.99786u^3 - 0.00349v^3 - 0.99990uv + 0.99757V_{stim}(t)$$

$$v' = 0.00550u - 0.00500v - 0.00500u^2$$

Exact coefficients:

$$u' = -0.20000u + 1.20000u^2 - 1.00000u^3 - 1.00000uv + 1.00000V_{stim}(t)$$

$$v' = 0.00550u - 0.00500v - 0.00500u^2$$

Figure 1. Example comparison of SINDy coefficient fits for the VF-4 model. Top: SINDy fit. Bottom: original model. $V_{stim}(t)$ represents the applied stimulus. Note that the v^3 SINDy coefficient in u' is the only value significantly different from the exact coefficient.

restricting the function library. For this model, the optimizer choice again was important—despite FHN-c varying little from the FHN model, only SSR with the Poly3 feature library successfully recovered the FHN-c model. This choice also maintained low sensitivity to changes in the α parameter, as it did for the FHN case.

Finally, both the VF-4 and VF-7 models were successfully identified using the same Poly3 and SSR combination that was found to be effective for FHN-c, once more displaying a desirable low sensitivity to the parameter α. Fig. 1 shows example results of fitting the VF-4 model.

We also examined the effects of altering the threshold (or α parameter for SSR) quantitatively, as shown in Fig. 2. In the upper plot, SSR shows a significantly lower mean squared error (MSE) than that of other optimizers for threshold values between 10^{-5} and 0.5 while also displaying minimal variation in the MSE across the range of α values tested. In comparison, the other methods achieve MSEs that differ by about ten orders of magnitude as their threshold values are varied, indicating that the threshold value must be chosen carefully for these methods.

In the lower plot of Fig. 2, the voltage and recovery variable columns were normalized to have a maximum magnitude of one before being passed to the STLSQ and SR3 optimizers, and the results are plotted against the SSR result from before. This case shows similar results, with SSR achieving low error across all α values tested. STLSQ reaches slightly lower MSE values than SSR for the lowest threshold values, but its higher sensitivity to the threshold parameter, similar to the sensitivity of SR3, means that the quality of the results can depend highly on the choice of the threshold value.

Finally, for cases with applied stimuli (period = 185 time units except for VF-7, with a period of 361 time units), stimulus terms were added to each model before applying the SINDy fit. Using the SSR optimizer with $\alpha = 10^{-5}$ yielded MSE values of $5.27617 \cdot 10^{-5}$ for FHN, $2.52412 \cdot 10^{-6}$ for FHN-c, $1.76467 \cdot 10^{-6}$ for VF-4, and $2.38930 \cdot 10^{-6}$ for VF-7. Note that for VF-4, this protocol produced alternans, as shown in Fig. 3. Identification was also successful under these more complex dynamics, with

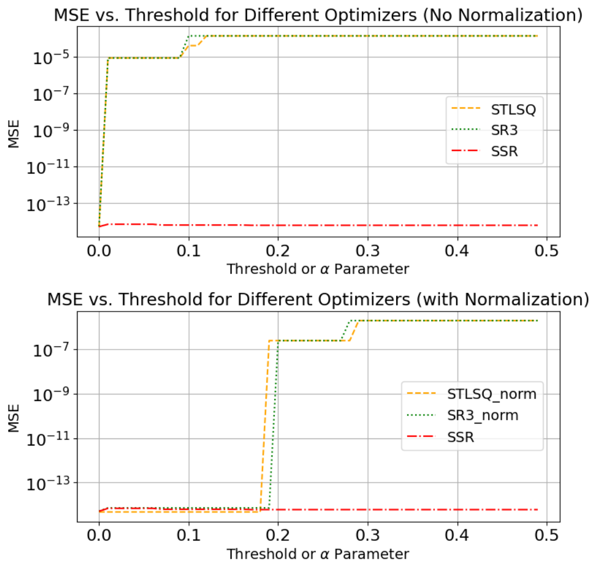

Figure 2. (Top) Comparison of Mean Squared Error (MSE) for different optimizers across threshold values ranging from 0 to 0.5 in increments of 0.01. (Bottom) The same comparison after normalizing the SINDy data before passing it to each optimizer.

the maximum difference in coefficient for any term having a magnitude of 0.0035 (maximum correct coefficient magnitude was 1.2).

4. Discussion

In this paper, we tested the use of SINDy to recover from data the specific differential equations of four two-variable models that have been used to describe cardiac action potentials, including during alternans. In fitting single action potentials, SSR performed better than the other optimizers tested across a variety of typical threshold values. Thus, SSR provides an advantage over STLSQ or SR3 for most use cases because threshold tuning is not required to obtain accurate results. Even when such tuning is done for STLSQ or SR3, very small thresholds are re-

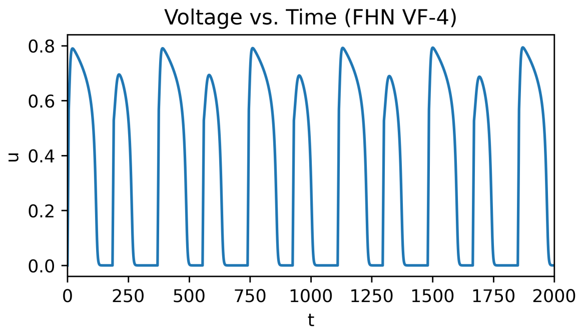

Figure 3. Voltage vs. time for the 4-parameter VF-4 variant. In this case, alternans is displayed under a stimulus of period 185, duration 5, and magnitude 0.12. Parameter values used are $\alpha = 0.2$, $\beta = 1.1$, $\epsilon = 0.005$, and $\mu = 1.0$.

quired to achieve the same level of accuracy as SSR, and such thresholds will introduce unnecessary model complexity by allowing more terms to remain in the final result. However, normalizing the data, as per the plot in the lower part of Fig. 2, improves the performance of the STLSQ and SR3 optimizers for thresholds below 0.2, bringing the MSE near machine precision. In fact, STLSQ outperforms SSR for thresholds between 0.1 and 0.2, likely due to the fast-slow nature of the FHN-based systems we tested. Whereas the voltage variable v ranges from 0 to 1 every AP before normalization, the recovery variable v takes on much smaller values due to the coefficient ϵ, whose value ranged between 10^{-2} and 10^{-3} for each SINDy fit.

Our MSE results for STLSQ at low thresholds initially seem to agree with SINDy benchmarking tests that found STLSQ to perform well enough in all cases to justify its use as the default optimizer in the PySINDy package [10]. In practice, however, we did not find this improvement to be significant—in fact, in our studies, SSR outperformed STLSQ on data containing time-dependent stimuli to the point where it remained the most promising optimizer choice for the two-variable cardiac models we tested.

5. Conclusion

The understanding of cardiac electrophysiology models stands to benefit greatly from data-driven methods. One particular method, SINDy, shows promise in identifying viable models that are both accurate and interpretable. In particular, SINDy successfully identified four two-variable cardiac models, including a case with alternans dynamics. Different optimizers displayed varying levels of effectiveness in identifying models, with SSR showing overall greater consistency than other options, including SR3 and STLSQ, when compared across model complexity and MSE metrics. Further study is needed to test SINDy's usefulness in discovering cardiac models directly from experimental data.

Acknowledgments

This study was supported by NIH grants 2R01HL143450 and T32GM142616.

References

[1] Noble D. A modification of the Hodgkin-Huxley equations applicable to Purkinje fibre action and pacemaker potentials. The Journal of Physiology 1962;160(2):317.

[2] Hodgkin AL, Huxley AF. A quantitative description of membrane current and its application to conduction and excitation in nerve. The Journal of Physiology 1952; 117(4):500.

[3] Brunton SL, Proctor JL, Kutz JN. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. Proceedings of the National Academy of Sciences 2016;113(15):3932–3937.

[4] Kaptanoglu AA, de Silva BM, Fasel U, Kaheman K, Goldschmidt AJ, Callaham J, Delahunt CB, Nicolaou ZG, Champion K, Loiseau JC, Kutz JN, Brunton SL. PySINDy: A comprehensive Python package for robust sparse system identification. Journal of Open Source Software 2022; 7(69).

[5] FitzHugh R. Impulses and physiological states in theoretical models of nerve membrane. Biophysical Journal 1961; 1(6):445–466.

[6] Rogers JM, McCulloch AD. A collocation-galerkin finite element model of cardiac action potential propagation. IEEE Transactions on Biomedical Engineering 1994; 41(8):743–757.

[7] Velasco-Perez HA. Methods for model reduction in cardiac dynamics. Ph.D. thesis, Georgia Intitute of Technology, 2022.

[8] Hindmarsh AC. ODEPACK, a systematized collection of ODE solvers. IMACS Transactions on Scientific Computation 1983;1:55–64.

[9] Welch C. SINDy-for-Cardiac-Electrophysiology, 2024. URL `https://github.com/cswelch/SINDy-for-Cardiac-Electrophysiology`.

[10] Kaptanoglu AA, Zhang L, Nicolaou ZG, Fasel U, Brunton SL. Benchmarking sparse system identification with low-dimensional chaos. Nonlinear Dynamics 2023; 111(14):13143–13164.

Address for correspondence:

Cole S. Welch, Elizabeth M. Cherry
756 West Peachtree Street NW, Atlanta, GA 30332
cwelch49@gatech.edu, elizabeth.cherry@gatech.edu